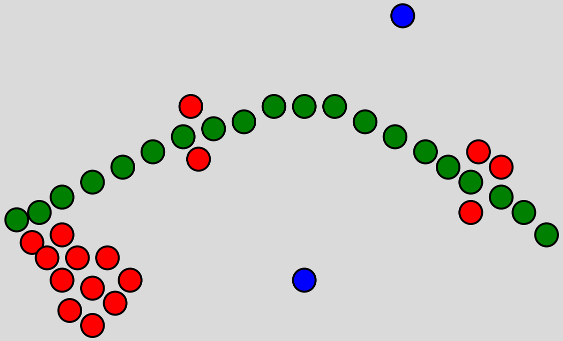
CCLRC
Rutherford Appleton Laboratory



Event Reconstruction

Initial *Track* / *Shower* Pattern Recognition

using *Artificial Neural Networks*

Costas Andreopoulos *<C.V.Andreopoulos@rl.ac.uk>*

MINOS Collaboration Meeting – Fermilab, Sep. 18, 2003

Outline

I have been using **Artificial Neural Networks** for **Neutrino Event Classification**.

(see my other talk in this Collaboration meeting)

Can they also be used for **Pattern Recognition** during **Event Reconstruction** ?

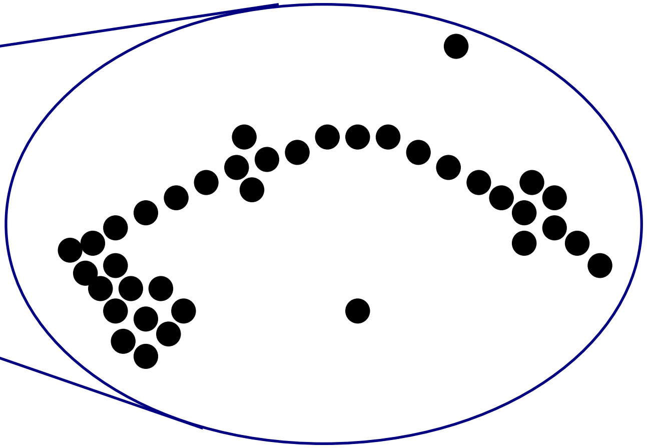
In this presentation:

- Motivation
- Quantification of topological patterns
- Neural Network training
- Neural Network performance
- Example events.
- Conclusions & further work

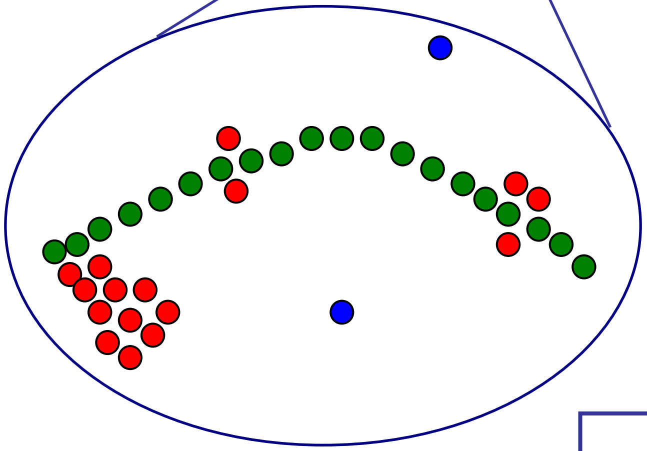
Initial thought & Goal

TASK: ***Event Reconstruction*** → identify showers & tracks

Why starting from this?



Can I start from this?

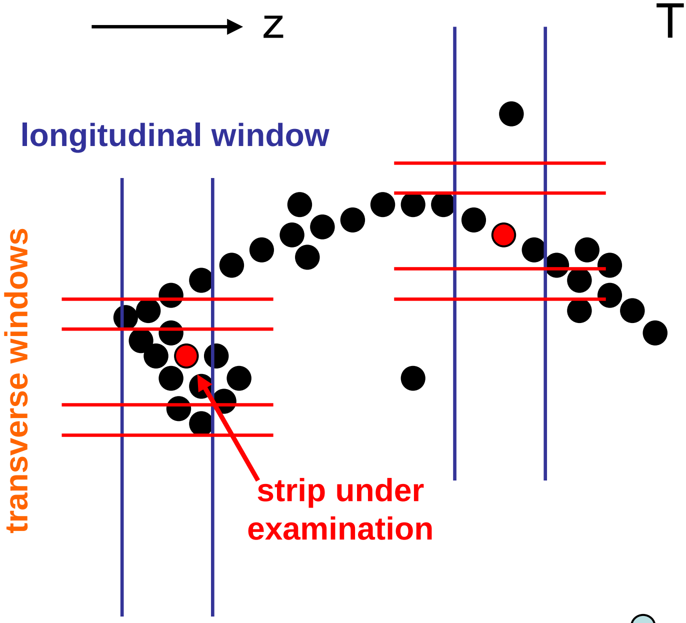


Is there a way to get an *‘almost’ correct answer* **very quickly and then refine it?**

COULD A NEURAL NETWORK WORK HERE?

- Very well suited for the task of **identifying topological patterns**…
- It should be **very fast** too... *You just evaluate the neural net function for the given input*

Quantifying the Topological Patterns



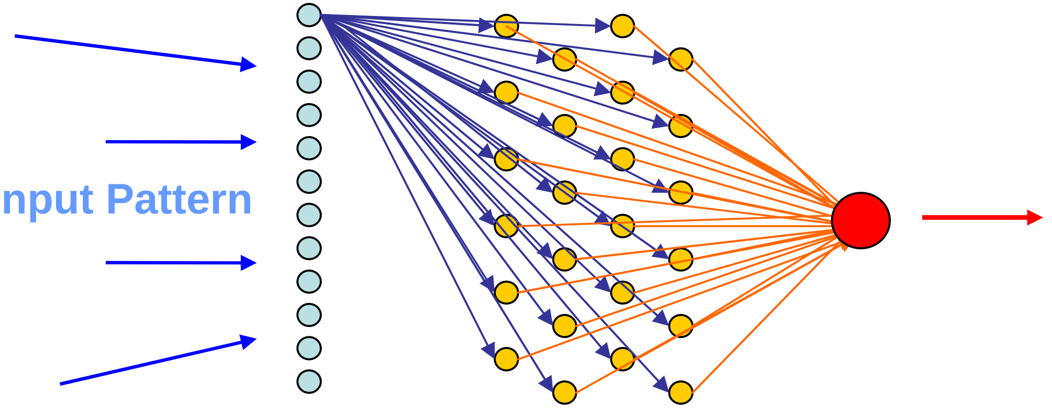
Topologically, a shower does look different than a track…

HOW do I quantify these differences?

Use the **number of other strips**, **the % of strips** and the **pulse height** in **transverse**, **longitudinal** and **radial** **windows** around the strip under examination

The quantified topological pattern can be used as input to an **Artificial Neural Network**

Neural Network Input Pattern



Neural Network Output

Answering to:
“Does this strip belong to a shower or a track?”

(For simplicity connections are shown for a single input node only)

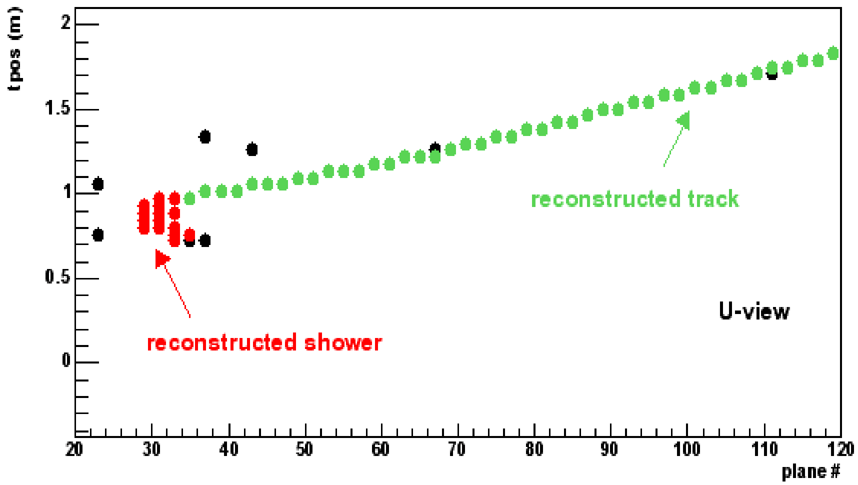
Input Layer

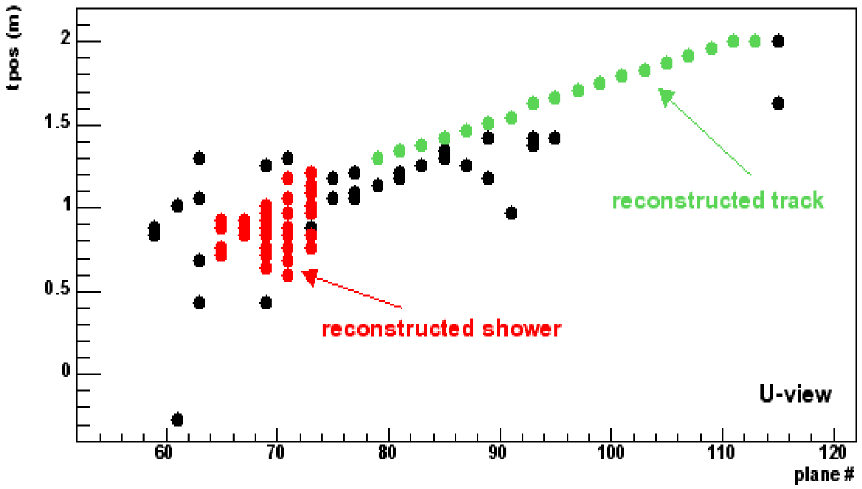
Hidden Layer

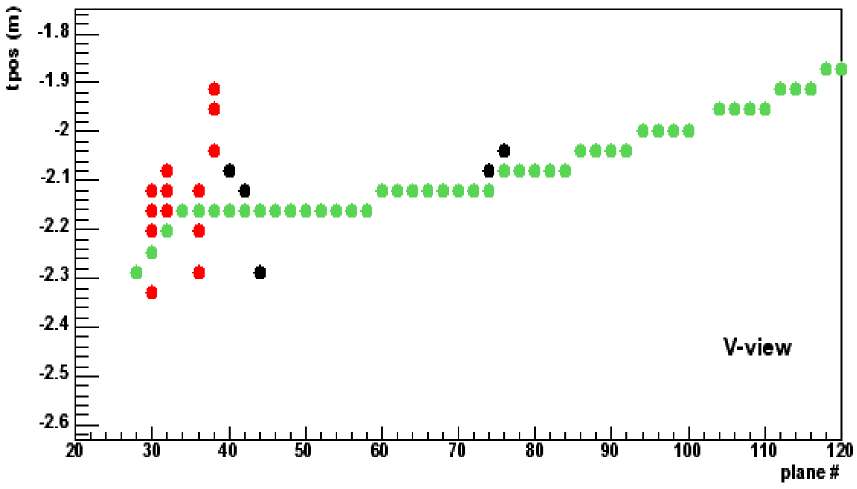
Output Layer

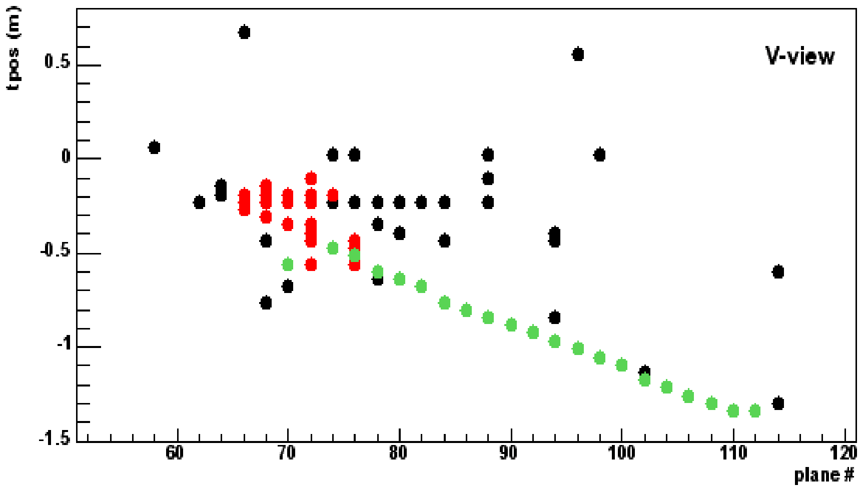
Where to find a training sample ?

Use MC events reconstructed with the SR packages – examples shown below:









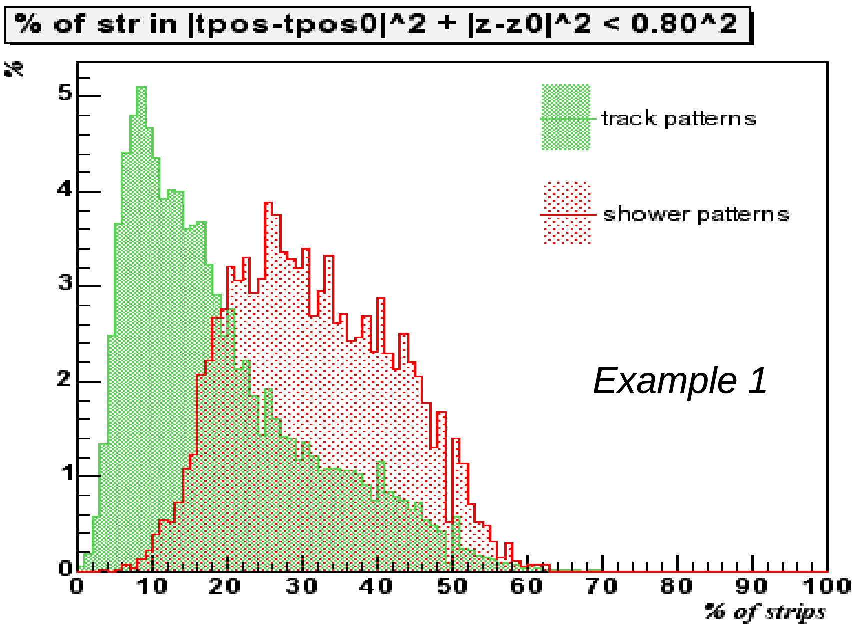
Neural Network Inputs

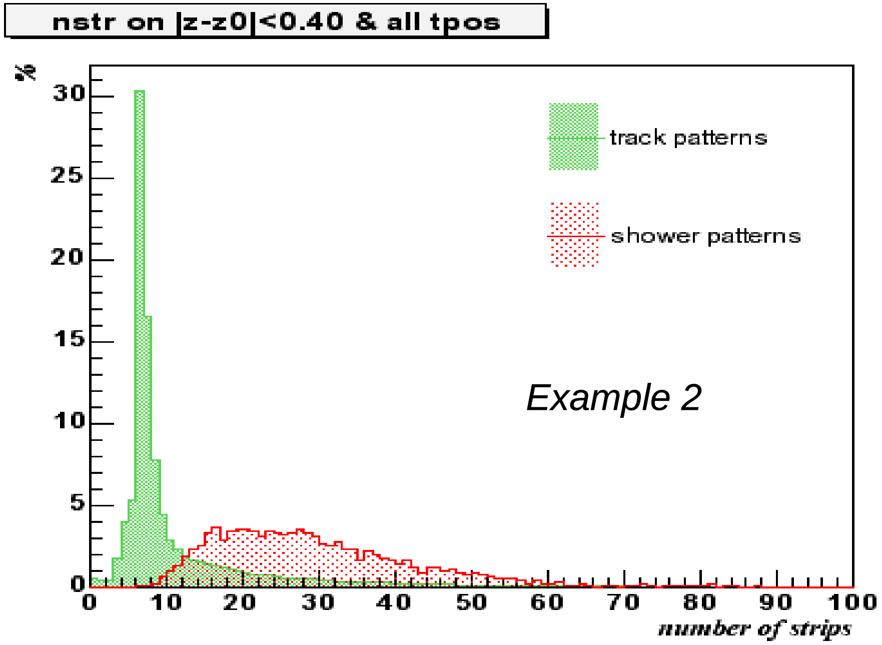
Right now I am using 72 input variables (!!) corresponding to different transverse, longitudinal & radial windows

There is certainly some redundancy… but the neural network **performs nice and fast**!

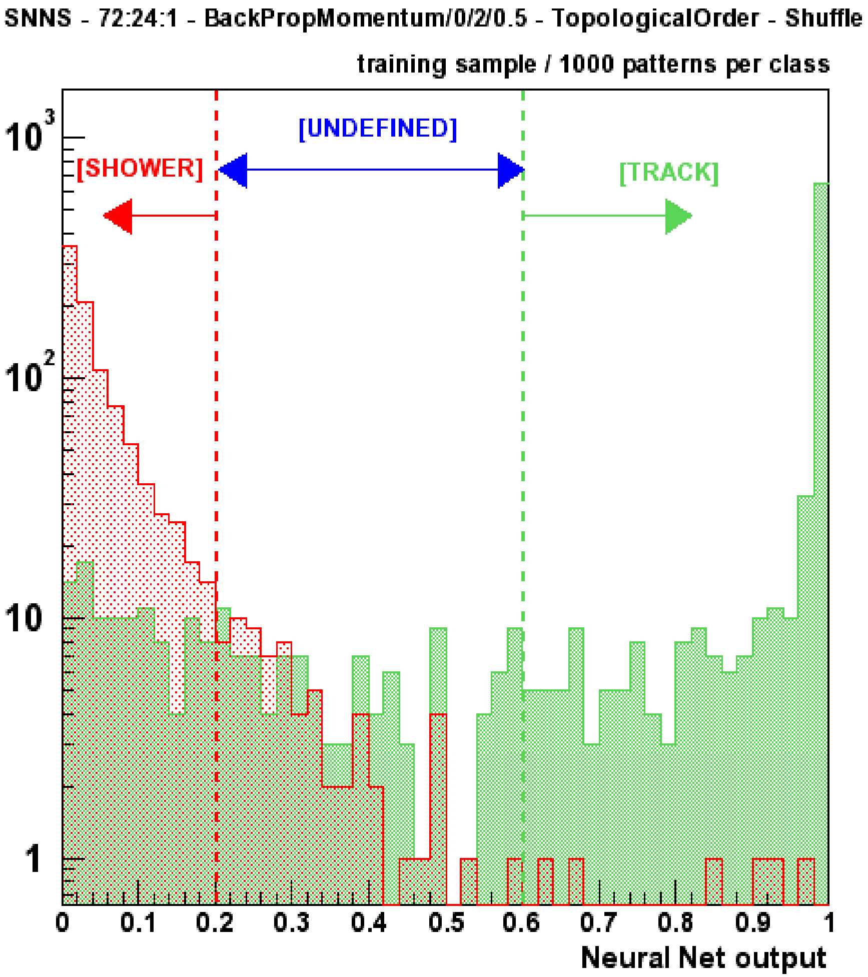
There is however some room for optimization by reducing the dimensionality.
In the next iteration I will try to do this playing with PCA *(Principal Components Analysis).*

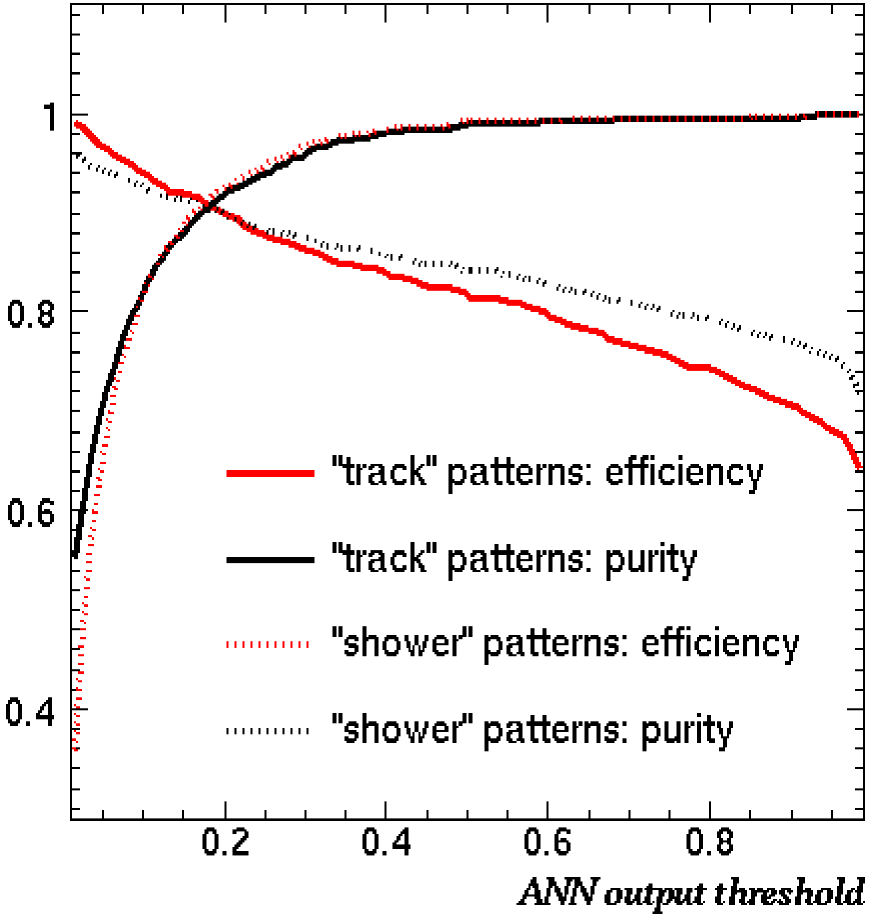
EXAMPLES OF INPUT VARIABLES:





Neural Network Performance





Results: *Example NEAR Det. MC Events filtered with neural net*

KEY for next plots:

🔴 **strips belonging to 'shower-like' formations**

🟢 **strips belonging to 'track-like' formations**

Quoted times:
on a 1.4 GHz Centrino (Dell D600) - 512 MB RAM

Quoted time is the time for:
looping over all strips, and for each one

- *compute the 72 neural net inputs*
- *evaluate the neural net function*
- *apply the threshold value*
- *plot on the 'event display'*

using interpreted C++ code (loon macros)

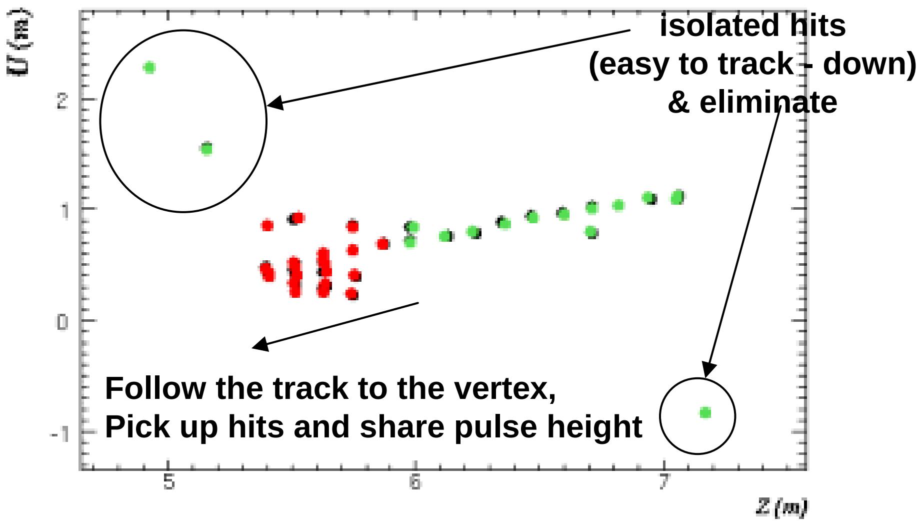
The **net computation time** using *compiled code* will be **much much less**

Note before exposed to critic:

Although pretty close,
*these **ARE NOT "final reconstructed events".***

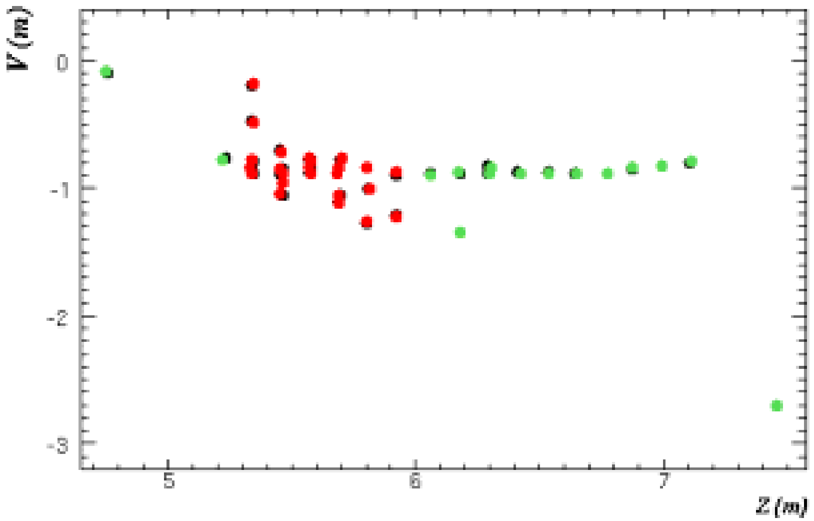
*They are just **"reconstruction seeds"** obtained in **SUB-SEC time scale** to aid subsequent reco.*

1st example -- CPU time: 0.44 sec



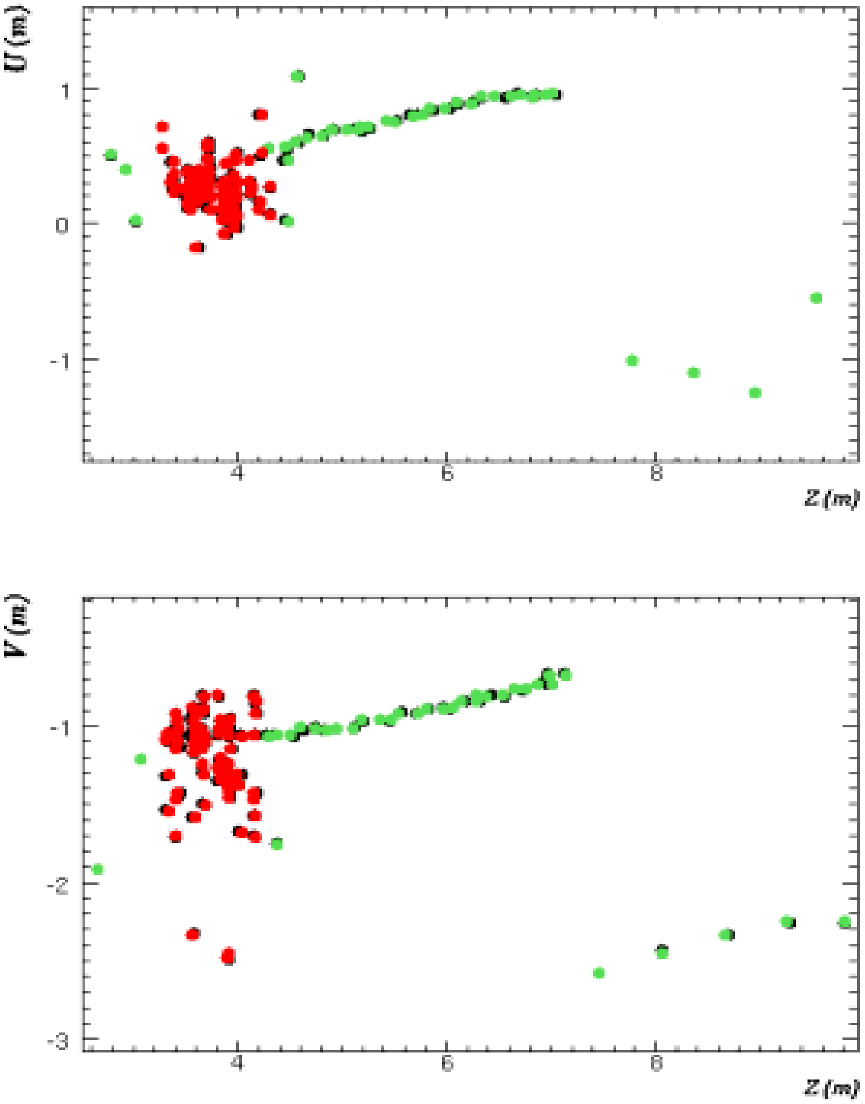
isolated hits (easy to track - down) & eliminate

Follow the track to the vertex, Pick up hits and share pulse height

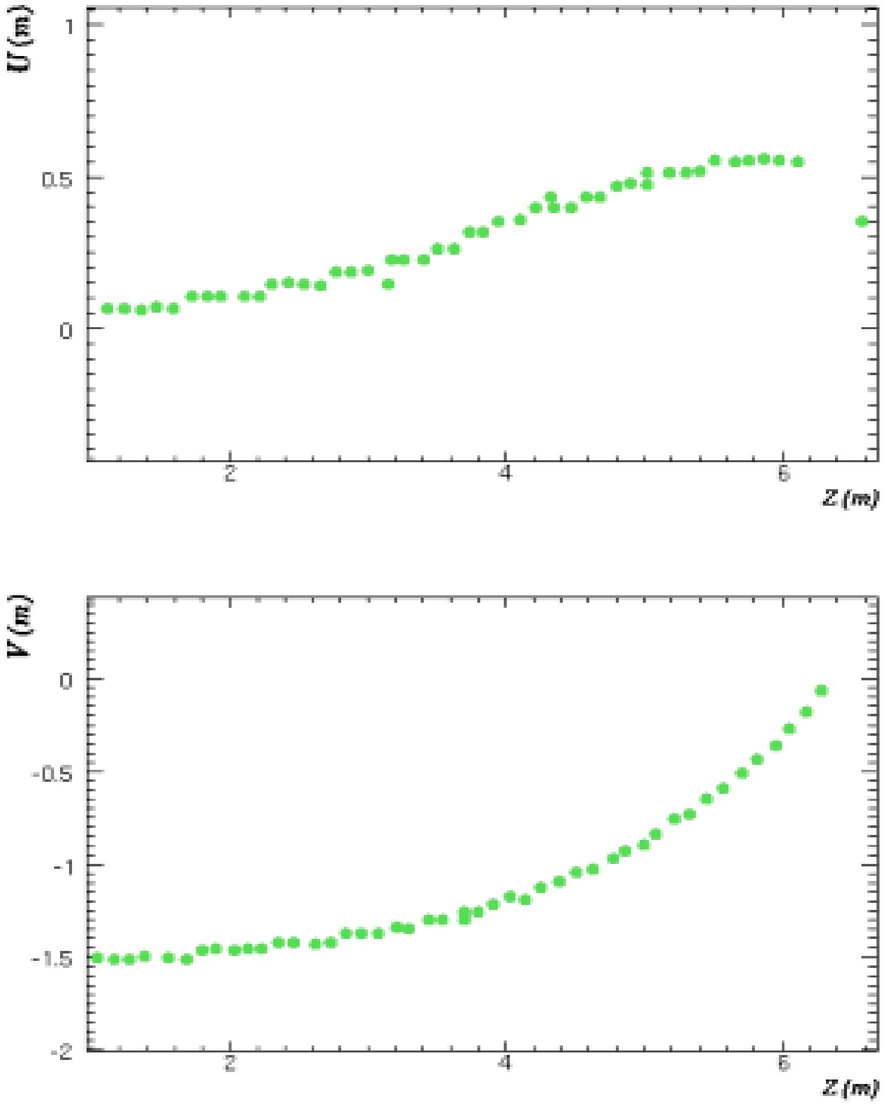


Results: *Example NEAR Det. MC Events filtered with neural net*

2nd example -- CPU time: 1.29 sec

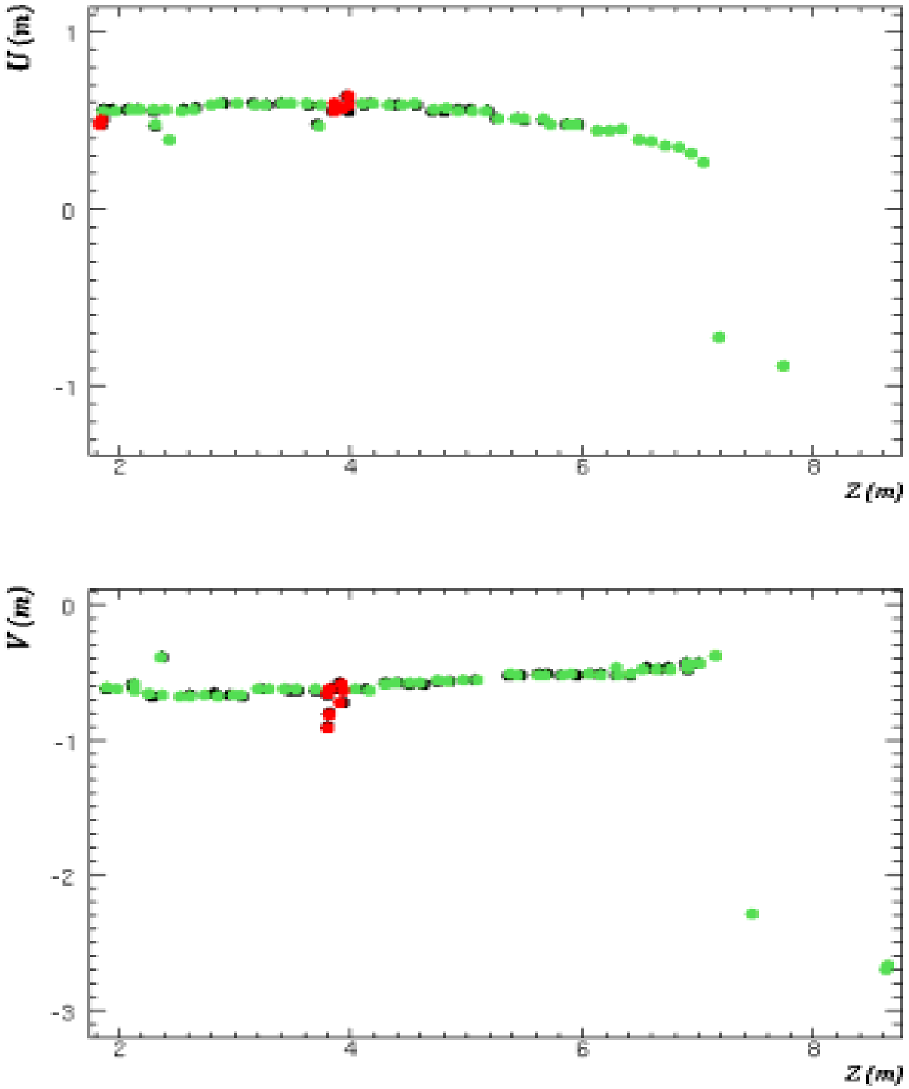


3rd example -- CPU time: 0.54 sec

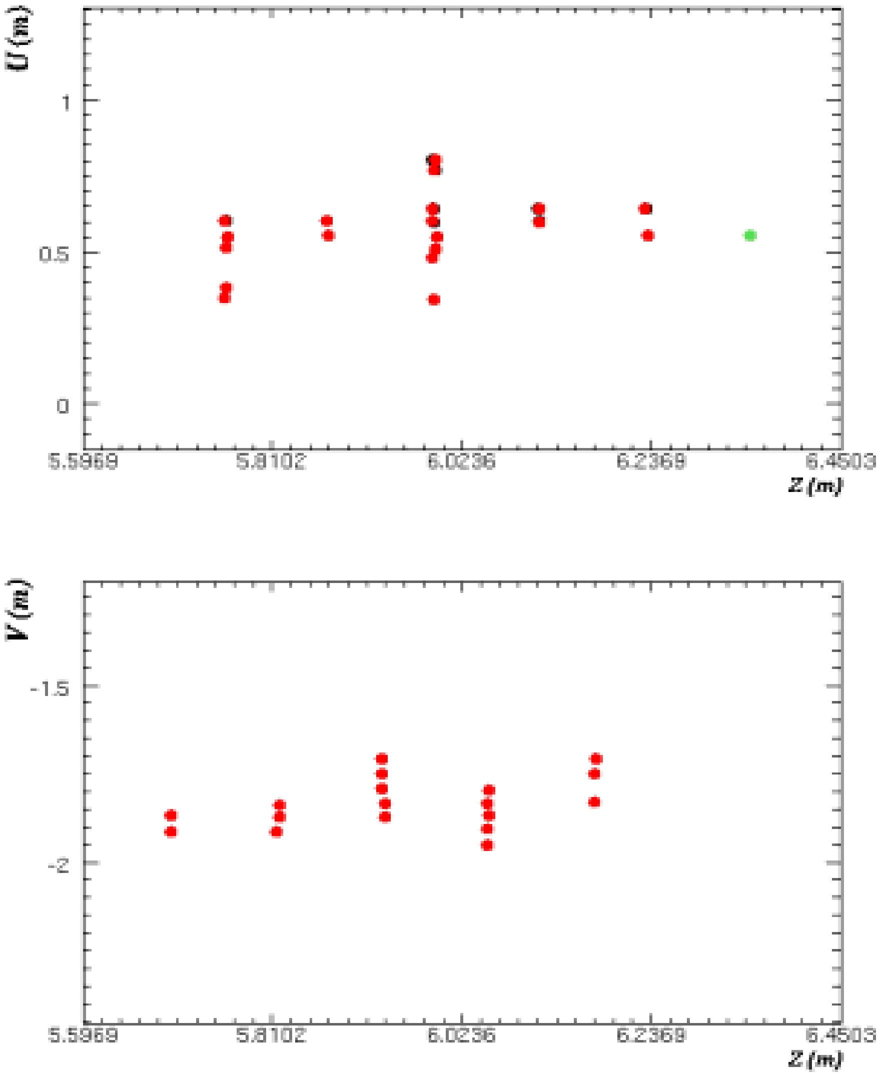


Results: *Example NEAR Det. MC Events filtered with neural net*

4th example -- CPU time: 0.65 sec

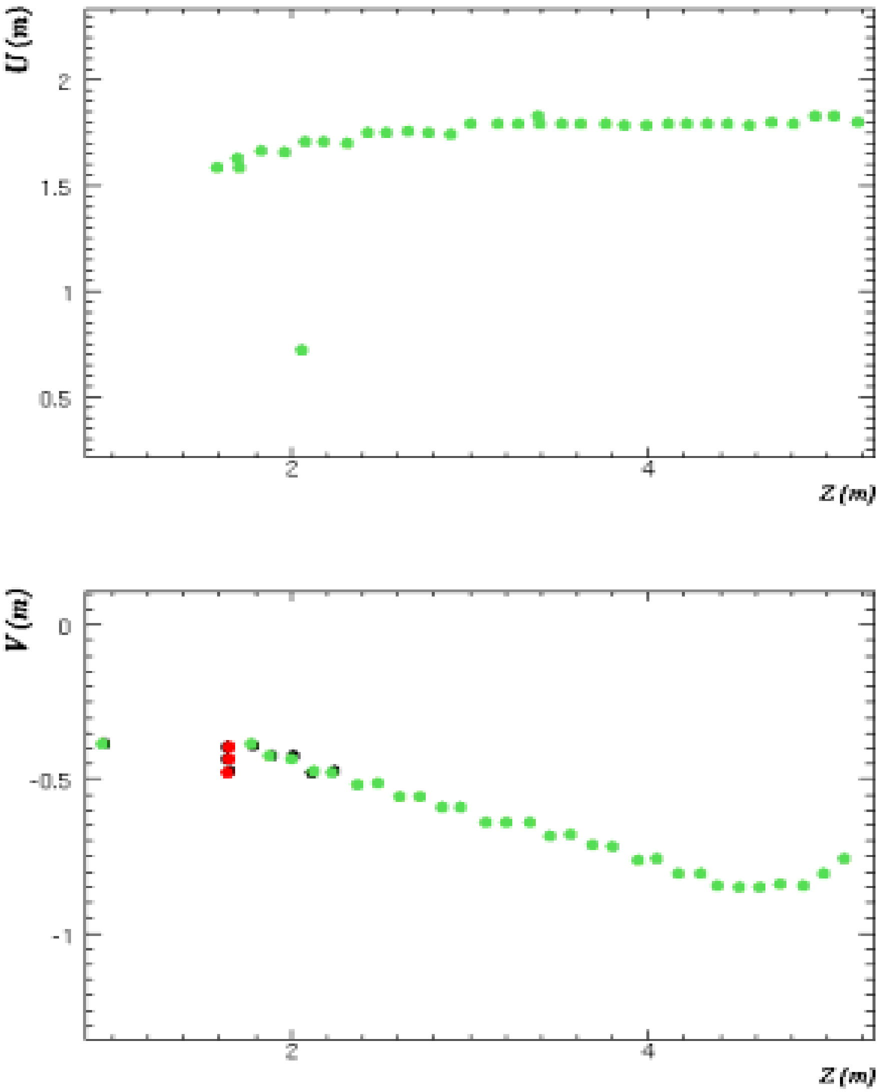


5th example -- CPU time: 0.22 sec

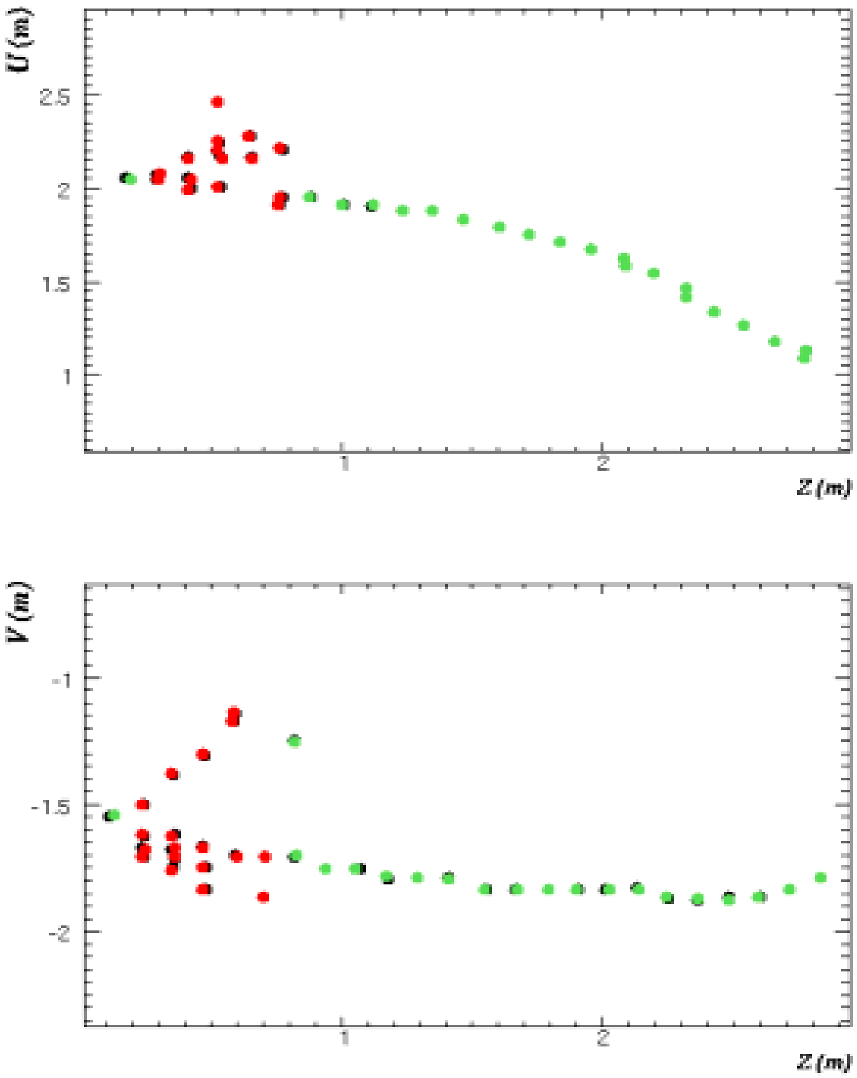


Results: Example NEAR Det. MC Events filtered with neural net

6th example -- CPU time: 0.38 sec

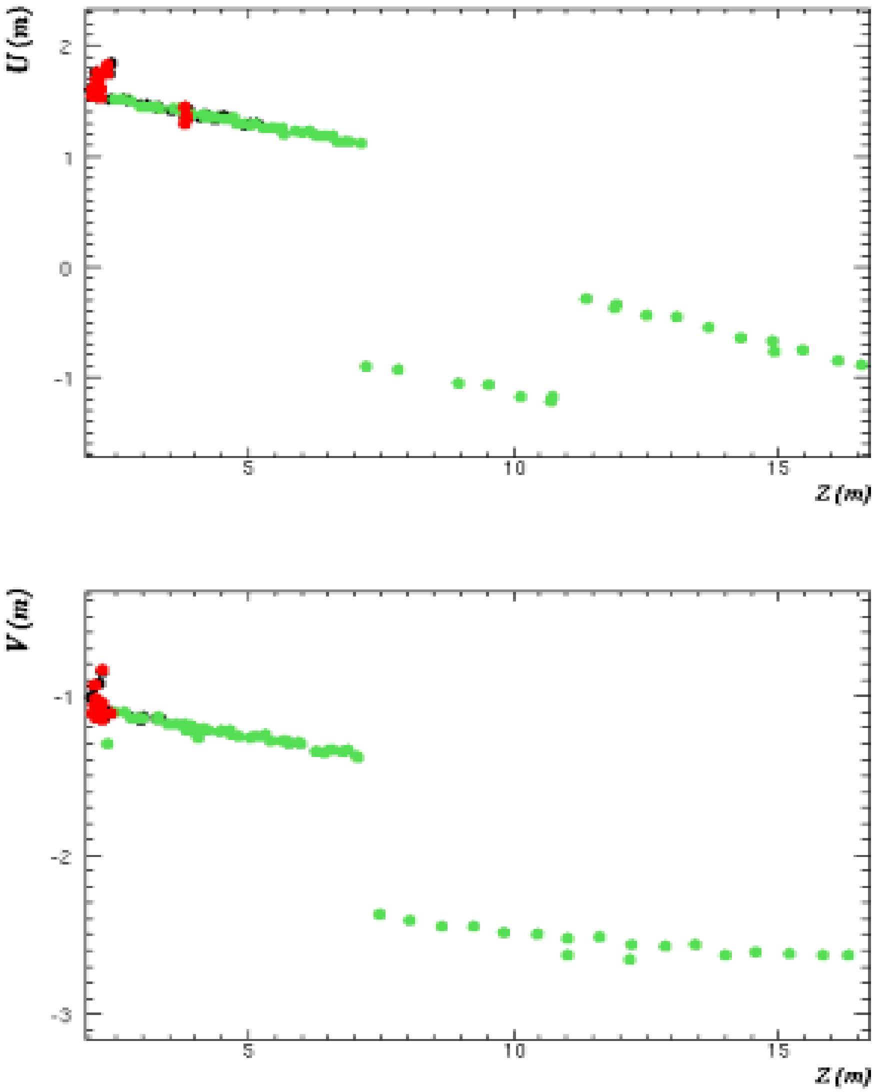


7th example -- CPU time: 0.42 sec

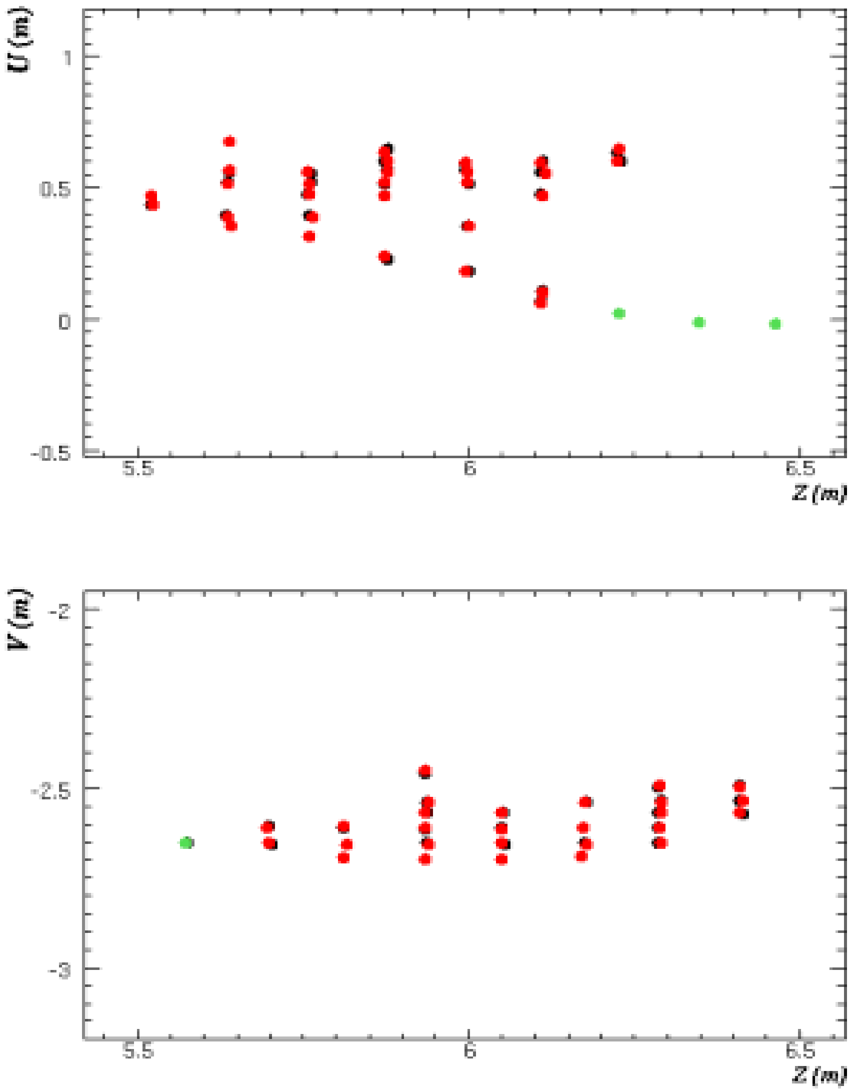


Results: *Example NEAR Det. MC Events filtered with neural net*

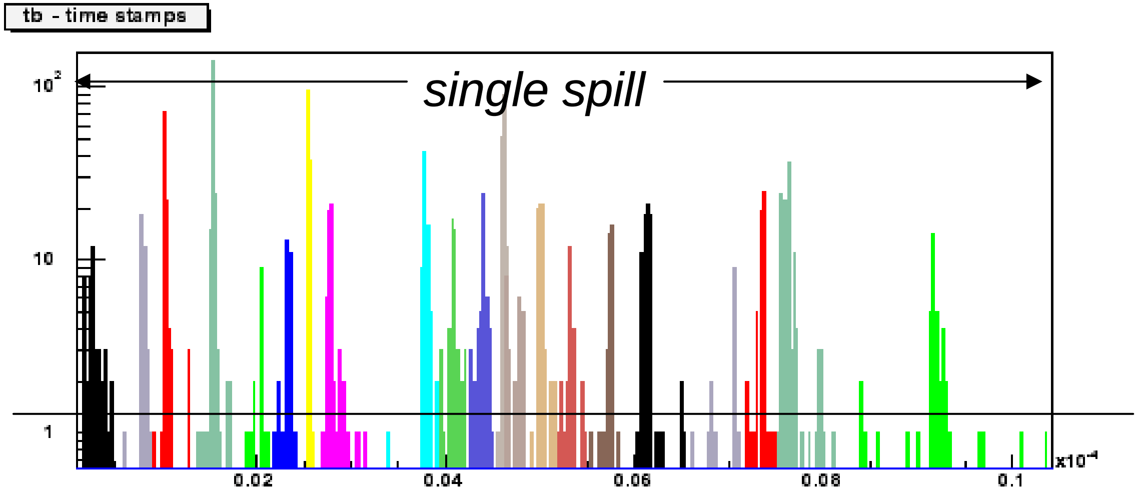
8th example -- CPU time: 0.88 sec



9th example -- CPU time: 0.35 sec

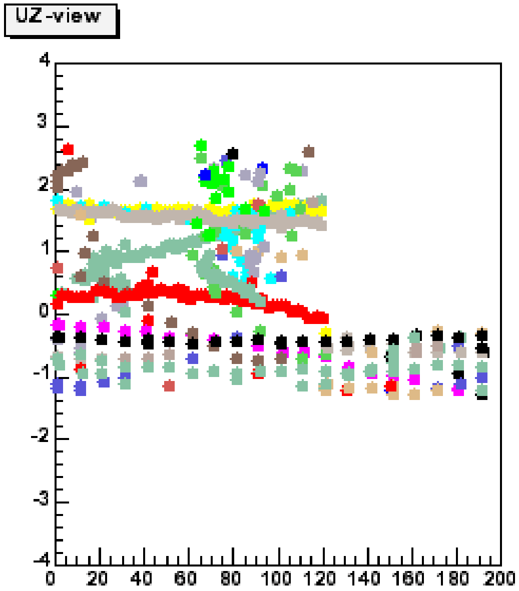


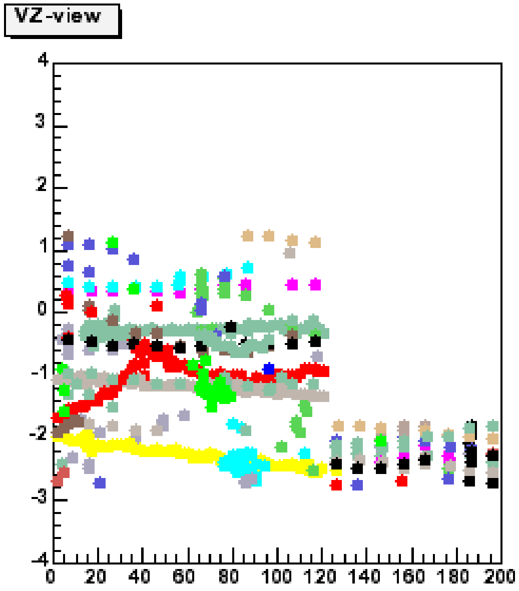
Event Slicing



Just a single slide on event slicing before finishing this presentation

It will be presented in detail at a later stage





Strategy:

- *Obtain slice seeds*
  - Take distribution of strip times
  - Apply a threshold to identify 'peaks'
  - Share the time between peaks based on relative peak pulse heights
- *Refine slices*
  - 3-D clustering in *(c*t, tpos, z) space*

Conclusions & Further work

- **An alternative reconstruction package *(AltReco)* is under development**

- **Framework-wise:**
  - *100% compliant with MINOS Offline framework (**an SR-clone**)*
  - "Similar" candidates, same output tree… no difference for the end user.

- **What already exists:**
  - **Event Slicing Algorithm:**
    - *an 'almost' correct first guess using **timing information** only, and then*
    - *refinement using **3-D clustering** in (c*time, z, transverse position) space.*
  - **Initial Track/Shower Pattern Recognition:**
    - *an 'almost' correct first guess using **Artificial Neural Networks**, and then*
    - *refinement (?)*

- **As it is based on 'almost' correct first guesses…**

  refinement does not last long: VERY FAST!

- **Next step:** Interfacing with existing trackers / write own ?
- **Next step:** Commit in CVS as soon as I feel confident with initial test and have documented its performance *(successes & failures)*